Title: **Sneak Peek at FORTRAN 2002**

Summary: The Power of today's machines is driving FORTRAN to new heights that will next be unveiled as the new FORTRAN 2002.

FORTRAN has long been considered the language for scientific and engineering computations. Because machines are ever more precisely engineered, the numerical controls that guide them must match these evolving hardware capabilities.

Simultaneously, developments in other programming languages are influencing FORTRAN's new directions. For example, elements of object-oriented programming that have been the cornerstone of other programming languages are finding their way into newer generations of FORTRAN. In addition, ISO has mandated interoperability between languages to ensure that this trend will continue. Thus, FORTRAN 2002 has moved in the direction of interoperability, particularly toward the C programming language.

In the past, interoperability between C and FORTRAN has been generally nonportable and often relatively inefficient. Because the two languages represent data differently, compilers had to insert executing instructions that converted the data representations into the appropriate form for the other language.

This time and expense is burdensome for several reasons. First, the popularity of C as a programming language makes it first choice for many involved in systems work or applications such as graphics, while the numerical precision and other fortes of FORTRAN make it an enduring choice for scientific and engineering applications. Making each language accessible to all users is certainly the better mode of operations. In addition, many operating systems and graphics libraries are available in C, but not in FORTRAN. Making the two languages interoperable saves engineers the bother of rewriting this software.

In FORTRAN 90/95 explicit interfaces were first introduced into the language to allow access to older FORTRAN 77 programs. These interfaces had the compiler check that you were calling the program correctly when you passed information between the older and newer programs. The previous implicit interfaces would not stop you from proceeding with mismatched data representations. The new explicit interfaces have the compiler check for a correct call, and then if correct, the program executes. An extension to the explicit interfaces for FORTRAN 2002 will permit the calling of C programs efficiently as well as the potential for solving other interoperability issues.

In FORTRAN 2002, the limitations of precision and range for real numbers in representing infinite mathematical constructs can be controlled by the Floating Point Exception Handling mechanism. This hardware feature communicates to the programmer that an exceptional condition has occurred arithmetically so that the programmer can handle the problem in a way that is appropriate to the problem. Practically speaking, this means the programs will not crash when confronted with overflow or underflow problems and other exceptional situations, but rather allow the software to handle these machine limitations.

This IEEE Exception Handling has wide applications for engineers. As an example, consider those studying fluid flow who need to define boundary conditions that tell how much fluid is flowing from one end. If they specify illegal boundary conditions that generate arithmetical anomalous behavior, they are likely to generate illegal output. Previously, if the number was to big (an overflow condition), the relationship between this illegal input and the error would not be clear. But with the new Floating Point

Exception Handling feature, the programmer can detect when illegal computations are being performed and so generate appropriate diagnostics describing the error. Actually, this capability has already been implemented by the Numerical Algorithms Group's, Downers Grove, IL, latest FORTRAN compiler.

One of the major inefficiencies in FORTRAN 90/95 is the use of pointers in lieu of direct addresses to data. While pointers give the advantage of flexibility (i.e., you can change your program to point to different data in different simulations) and eliminate the inefficiency caused by moving large amounts of data around in your machine, they come with the inherent problem of being difficult to optimize. Pointers can be twice as slow as allocatable components. This is because pointers, be definition, have one level of indirection built in, and compilers are unable to determine where the object is until very late in the execution of a calculation. Allocatable components avoid this extra level of indirection.

The developers recognize that dynamic data structures can be created with allocatable components in many cases. Therefore, the introduction of allocatable components supports FORTRAN 2002's goal of creating efficient numeric code. New capabilities are being built into the language with an eye toward optimization and maintaining FORTRAN's niche in the numerical controls needed in most engineering applications.

By Brian Smith
Smith is a member of the board of directors of the Numerical Algorithms Group and professor of science at the University of New Mexico.

Originally published in R&D Magazine, March 2002 (www.rdmag.com).

Numerical Algorithms Group
www.nag.com
infodesk@nag.com

